Instituto Politécnico Nacional

ESCUELA SUPERIOR DE COMPUTO

ESCOM

Compiladores

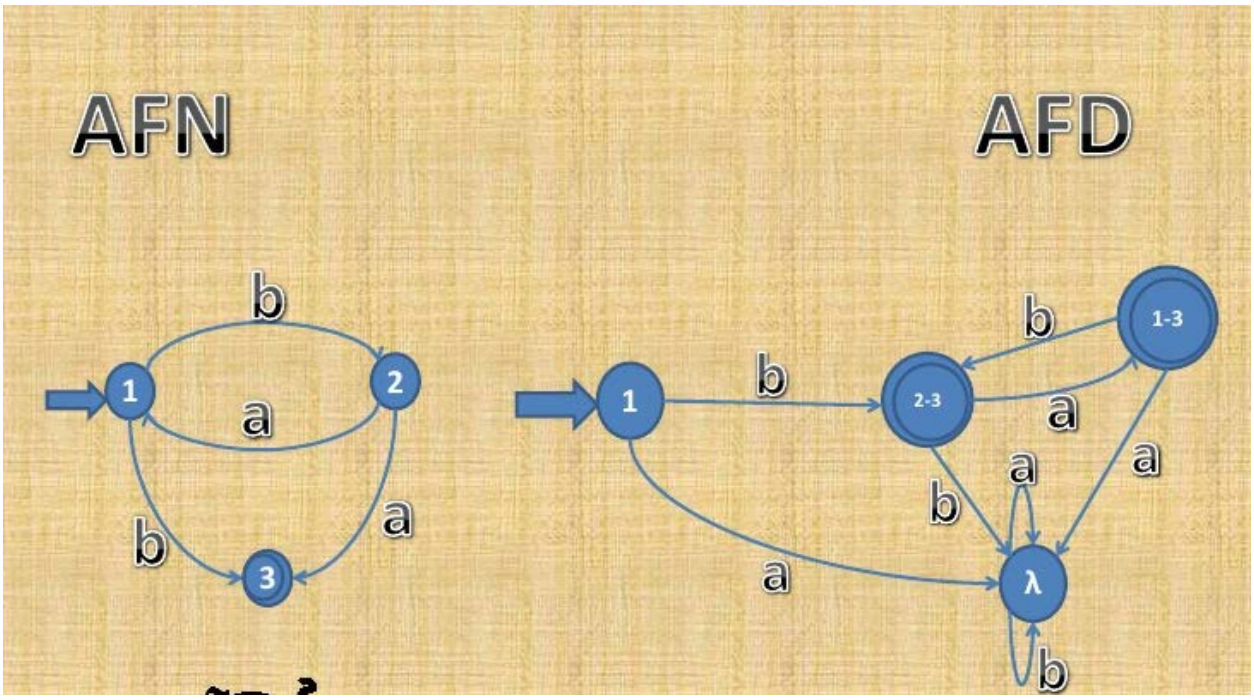
Práctica 1

AFD & AFN

PROFESOR: RAFAEL NORMAN SAUCEDO DELGADO

ALUMNO: DANIEL MURGUÍA JIMÉNEZ

GRUPO: 3CV6



Introducción

Los autómatas son modelos que describen maquinas de estados finitas las cuales, dada una entrada de símbolos, saltan de estado, a través de una función de transición. Dentro de la teoría de autómatas, dependiendo de la función de transición que el autómata utiliza, puede ser clasificado como Automata Finito Determinista(AFD) ó Automata Finito No Determinista (AFN).

Desarrollo

Para el desarrollo de esta práctica se pidió crear dos clases utilizando el paradigma orientado a objetos, cada una correspondiendo al modelo de un AFD y un AFN respectivamente, las cuales deberían de cumplir con dos requerimientos a su vez, siendo el primero poder visualizar los autómatas en un formato de archivo .af, que sería descrito siguiendo una serie de reglas:

1. Los símbolos del alfabeto que pueden usar para el autómata son: a,c,d,...,x,y,z. (sin acentos, sin la ñ), 26 símbolos en total.
2. Se usa E para detonar a épsilon.
3. Las etiquetas de los estados son números enteros positivos: 1,2,3,4,...(no hay estado cero)
4. Cada línea define una transición del autómata, en el formato, estado->estado, simbolo
5. La primera línea indica al estado inicial: inicial:estado
6. La segunda línea indica a los estados finales: finales:estado, estado, estado, estado
7. Las lineas siguientes, cada una, indican una transición.

El segundo requerimiento a cumplir es que las dos clases implementaran una serie de métodos definidos.

Para el desarrollo se utilizó el lenguaje de programación JAVA, el cuál soporta el paradigma orientado a objetos. Para el funcionamiento de la práctica se crearon 4 clases, siendo dos de estas las clases AFD y AFN, la clase Transición y la clase main para poder ejecutarlo. Dentro de la clase Transición se describió el modelo de una transición ocupada por el automata, de manera que sea posible manipularlas por los métodos de las clases de autómatas.

Las clases AFD y AFN fueron construidas de manera similar, puesto a que la diferencia entre estas radica en las transiciones, Las clases cuentan con su estado inicial y sus estados finales, así como su conjunto de transiciones que está manejado por un ArrayList de Objetos Transición; manejar el conjunto de transiciones de esta manera, facilita la manera en que podemos obtener la información acerca de estas, así como su manipulación. Para crear la definición del autómata, se utilizo la lectura de archivos, extrayendo la información mediante expresiones regulares que identificaban los estados inicial y finales y las transiciones; Tras reconocer esta información del archivo se procede a extraer los diversos valores de las cadenas encontradas mediante el objeto Matcher de JAVA que nos permite descomponer el

texto obtenido por la expresión regular en grupos, procediendo a guardarlas en variables del propio objeto..

En esta parte de desarrollo se presentaron problemas relacionados con el manejo de archivos puesto a que no cuento con conocimientos y práctica suficiente acerca del manejo de archivos en java lo que ocasionó que se presentaran errores y excepciones relacionados a la apertura y lectura de los archivos. Esto retraso el desarrollo de manera que no se pudo acabar en su totalidad pues no cuenta con los métodos para guardar la definición del autómata en un archivo, así como los métodos de validación de autómata y de generación de cadenas aceptadas. En consecuencia, no se cumplieron los requerimientos de la práctica, quedando inconclusa.

Código de clase AFD

```
1.  import java.io.*;
2.  import java.util.ArrayList;
3.  import java.util.Arrays;
4.  import java.util.regex.Matcher;
5.  import java.util.regex.Pattern;
6.
7.  public class AFD {
8.      private int inicial;
9.      private int [] finales;
10.     private ArrayList<Transicion> transiciones = new ArrayList<Transicion>();
11.     private File file;
12.     private FileWriter fw;
13.     private FileReader fr;
14.     private BufferedReader br;
15.
16.     public AFD() {
17.     }
18.
19.     public void cargar_desde(String nombre){
20.         this.file = new File(nombre);
21.     }
22.
23.     private void cargar_transiciones(){
24.         try{
25.             this.fr = new FileReader(this.file);
26.             this.br = new BufferedReader(this.fr);
27.             String linea;
28.             int inicio, fin;
29.             char c;
30.             Pattern pat = Pattern.compile("(\\d+)->(\\d+),([a-zE])");
31.             linea = br.readLine();
32.             while (linea != null){
33.                 Matcher matcher = pat.matcher(linea);
34.                 if (matcher.find()){
35.                     inicio = Integer.parseInt(matcher.group(1));
36.                     fin = Integer.parseInt(matcher.group(2));
37.                     c = matcher.group(3).charAt(0);
```

```
                    agregar_transicion(inicio,fin,c);
                }
            }
            fr.close();
            br.close();
        }catch(IOException e){}
    }

    public void guardar_en(String nombre){

    }
    public void agregar_transicion(int inicio, int fin, char simbolo){
        Transicion transicion = new Transicion(inicio,fin,simbolo);
        transiciones.add(transicion);
    }
    public void eliminar_transicion(int inicio, int fin, char simbolo){
        Transicion traux = new Transicion(inicio,fin,simbolo);
        int indice = 0;
        for(Transicion transicion : this.transiciones){
            if (transicion.equals(traux)){
                transiciones.remove(indice);
            }
            indice ++;
        }
    }
    public void obtener_inicial(){
        try{
            this.fr = new FileReader(this.file);
            this.br = new BufferedReader(this.fr);
            String linea = br.readLine();
            Pattern pat = Pattern.compile("([a-zA-z]+)(\\d+)");
            Matcher matcher = pat.matcher(linea);
            matcher.find();
            this.inicial = Integer.parseInt(matcher.group(2));
            fr.close();
        }catch (FileNotFoundException f){

        }catch (IOException e){}
    }
    public void obtener_finales(){
        try{
            this.fr = new FileReader(this.file);
            this.br = new BufferedReader(this.fr);
            String linea;
            int indice = 0;
            String [] aux;
            int [] nums;
            Pattern pat = Pattern.compile("([a-zA-z]+):(.\\s*\\d+)+");
            for (int i = 0; i <= 1; i++){
                linea = br.readLine();
                if (i == 1){
                    Matcher matcher = pat.matcher(linea);
                    matcher.find();
                    this.finales = Arrays.stream(matcher.group(2).split("\\s")).mapToIn
t(Integer::parseInt).toArray();
                }
            }
            fr.close();
            br.close();}catch(IOException e){}
    }
```

```
97.     public void establecer_inicial(int estado){
98.
99.     }
100.            public void establecer_final(int estado){
101.                int [] aux = Arrays.copyOf(this.finales,this.finales.length+1);
102.                aux[aux.length-1] = estado;
103.                this.finales = aux;
104.            }
105.            public boolean esAFN(){
106.                for(Transicion transicion : this.transiciones){
107.                    if (transicion.getSimbolo() == 'E'){
108.                        return true;
109.                    }
110.                }
111.                return false;
112.            }
113.            public boolean esAFD(){
114.                for(Transicion transicion : this.transiciones){
115.                    if (transicion.getSimbolo() == 'E'){
116.                        return false;
117.                    }
118.                }
119.                return true;
120.            }/*
121.            public boolean acepta(String cadena){
122.
123.            }
124.            public String generar_cadena(){
125.
126.            }*/
127.
128.        }
```

Código de clase AFN

```
1.  import java.io.*;
2.  import java.util.ArrayList;
3.  import java.util.Arrays;
4.  import java.util.regex.Matcher;
5.  import java.util.regex.Pattern;
6.
7.  public class AFN {
8.      private int inicial;
9.      private int [] finales;
10.     private ArrayList<Transicion> transiciones = new ArrayList<Transicion>();
11.     private File file;
12.     private FileWriter fw;
13.     private FileReader fr;
14.     private BufferedReader br;
15.
16.     public AFN() {
17.     }
18.
19.     public void cargar_desde(String nombre){
20.         this.file = new File(nombre);
21.     }
22.
23.     private void cargar_transiciones(){
24.         try{
```

```
            this.fr = new FileReader(this.file);
            this.br = new BufferedReader(this.fr);
            String linea;
            int inicio, fin;
            char c;
            Pattern pat = Pattern.compile("(\\d+)->(\\d+),([a-zE])");
            linea = br.readLine();
            while (linea != null){
                Matcher matcher = pat.matcher(linea);
                if (matcher.find()){
                    inicio = Integer.parseInt(matcher.group(1));
                    fin = Integer.parseInt(matcher.group(2));
                    c = matcher.group(3).charAt(0);
                    agregar_transicion(inicio,fin,c);
                }
            }
            fr.close();
            br.close();
        }catch(IOException e){}
    }

    public void guardar_en(String nombre){

    }
    public void agregar_transicion(int inicio, int fin, char simbolo){
        Transicion transicion = new Transicion(inicio,fin,simbolo);
        transiciones.add(transicion);
    }
    public void eliminar_transicion(int inicio, int fin, char simbolo){
        Transicion traux = new Transicion(inicio,fin,simbolo);
        for(Transicion transicion : this.transiciones){
            if (transicion.equals(traux)){
                transiciones.remove(transicion);
            }
        }
    }
    public void obtener_inicial(){
        try{
        this.fr = new FileReader(this.file);
        this.br = new BufferedReader(this.fr);
        String linea = br.readLine();
        Pattern pat = Pattern.compile("([a-zA-z]+)(\\d+)");
        Matcher matcher = pat.matcher(linea);
        matcher.find();
        this.inicial = Integer.parseInt(matcher.group(2));
        fr.close();
        }catch (FileNotFoundException f){

        }catch (IOException e){}
    }
    public void obtener_finales(){
        try{
        this.fr = new FileReader(this.file);
        this.br = new BufferedReader(this.fr);
        String linea;
        int indice = 0;
        String [] aux;
        int [] nums;
        Pattern pat = Pattern.compile("([a-zA-z]+):(.\\s*\\d+)+");
        for (int i = 0; i <= 1; i++){
```

```
            linea = br.readLine();
            if (i == 1){
                Matcher matcher = pat.matcher(linea);
                matcher.find();
                this.finales = Arrays.stream(matcher.group(2).split("\\s")).mapToInt(Int
    eger::parseInt).toArray();
            }
        }
        fr.close();
        br.close();}catch(IOException e){}
    }
    public void establecer_inicial(int estado){

    }
    public void establecer_final(int estado){
        int [] aux = Arrays.copyOf(this.finales,this.finales.length+1);
                aux[aux.length-1] = estado;
                this.finales = aux;
            }
            public boolean esAFN(){
                for(Transicion transicion : this.transiciones){
                    if (transicion.getSimbolo() == 'E'){
                        return true;
                    }
                }
                return false;
            }
            public boolean esAFD(){
                for(Transicion transicion : this.transiciones){
                    if (transicion.getSimbolo() == 'E'){
                        return false;
                    }
                }
                return true;
            }/*
            public boolean acepta(String cadena){

            }
            public String generar_cadena(){

            }*/

            }
```

Código de clase Transición

```
public class Transicion {
    private int inicio, fin;
    private char simbolo;
    public Transicion(){
        inicio = 0;
        fin = 0;
        simbolo = 'E';
    }
    public Transicion(int inicio, int fin, char simbolo){
        this.inicio = inicio;
        this.fin = fin;
        this.simbolo = simbolo;
    }
```

```
14.
15.     public int getInicio() {
16.         return inicio;
17.     }
18.
19.     public int getFin() {
20.         return fin;
21.     }
22.
23.     public char getSimbolo() {
24.         return simbolo;
25.     }
26.
27.    public boolean equals(Transicion tr) {
28.         int in = tr.getInicio();
29.         int fn = tr.getFin();
30.         char c = tr.getSimbolo();
31.         if(inicio == in && fin == fn && simbolo == c){
32.             return true;
33.         }else{
34.             return false;
35.         }
36.     }
37. }
```